Introduction au calcul scientifique avec Matlab
TP4
CP1 – 2019/2020
Fekak Fatima-Ezzahra

Exercice 1.

1. Écrire un script qui contient les instructions suivantes :
- Tirer 100 couples de points (x, y) aléatoirement dans le carré [0, 1] × [0, 1].
- Représenter le nuage de points obtenus dans une fenêtre graphique.
- Calculer le centre de gravité G du nuage de points.
- Ajouter en rouge au nuage de points le centre de gravité.
- Sauvegarder la figure sous le nom nuage.fig.
- Sauvegardez x, y et le centre de gravité G dans un fichier binaire sauve.mat.

2. Dans Matlab, avec la commande clear, effacez x, y et G. Puis, chargez le fichier sauve.mat afin de les récupérer. Vérifiez la récupération.

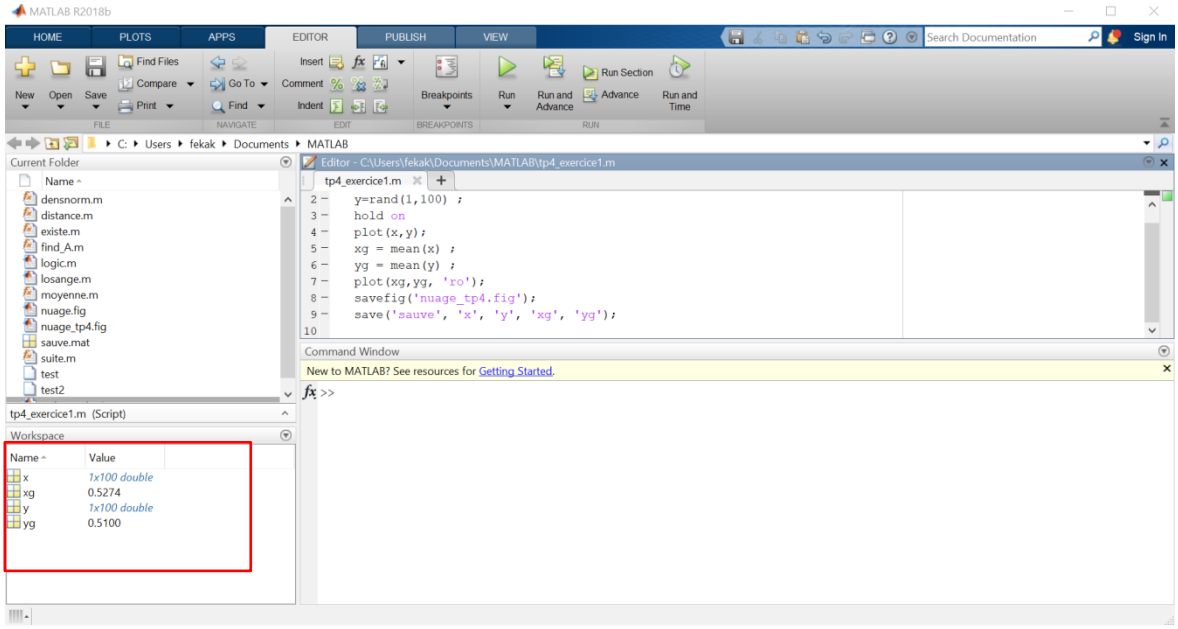
Réponse :

1.- Créer un fichier tp4_exercice.m
- Y mettre les instructions suivantes (ces instructions peuvent être récupérées du tp3)

```
x=rand(1,100) ;
y=rand(1,100) ;
hold on
plot(x,y);
xg = mean(x) ;
yg = mean(y) ;
plot(xg,yg, 'ro');
savefig('nuage_tp4.fig');
save('sauve', 'x', 'y', 'xg', 'yg');
```

2. Pour supprimer toutes les variables du workspace :
>> clear all ;

Pour ouvrir le fichier sauve.mat préalablement enregistré :
>> load sauve.mat => charge toutes les variables enregistrées dans ce fichier (voir dernière ligne du script ci-dessus).
On peut remarquer que ces variables s'affichent dans la sous-fenêtre « Workspace » en bas à gauche de votre fenêtre Matlab (voir le cadran rouge sur l'image ci-dessous)



Pour vérifier la récupération, il suffit d'appeler les variables par leurs noms d'origine :

```
>> xg
>> yg
>> x
>> y
```

Exercice 2.

1. Écrire une fonction Matlab **GenereMatrice** générant une matrice d'ordre quelconque avec des éléments aléatoires. La fonction aura comme paramètres d'entrée le nombre de lignes n, le nombre de colonnes m et un coefficient multiplicateur Amax.

2. Écrire une fonction Matlab **GenereSysteme** construisant des systèmes linéaires aléatoires, et les résolvant (lorsque c'est possible). La fonction aura comme paramètres d'entrée la taille du problème. Et en paramètres de sortie un message qui indique si le système a une solution ou non. En cas de solution afficher également la valeur de la solution.

Réponse :

1. - Créer un fichier GenereMatrice.m
   - Y mettre le code suivant :

```
function A = GenereMatrice(m,n,Amax)
%GENEREMATRICE génère une matrice d'ordre quelconque mxn avec des éléments aléatoires
%   les éléments de la matrice A  sont compris entre 0 et Amax
A = Amax*rand(m,n);
end
```

Pour vérifier qu'il n'y a pas d'erreur dans la fonction, on tape par exemple :
>> A = GenereMatrice(10,11,14)

2. - Créer un fichier GenereMatrice.m
   - Y mettre le code suivant :

```
function [A,b,x,msg] = GenereSysteme(m,Amax,bmax)
%GenereSysteme construit des systèmes linéaires aléatoires de taille met et les résolve (lorsque c'est possible).
%   les éléments de la matrice A  sont compris entre 0 et Amax
%   les éléments de du second membre b  sont compris entre 0 et bmax
%   msg est un message qui indique si le système a une solution ou pas
%   en cas de solution, x contient la valeur de la solution
A = GenereMatrice(m,m,Amax);
b = bmax*rand(m,1);
if det(A)~=0
msg = 'il y a une solution';
x = A\b;
else
x = 0;
msg = 'pas de solution';
end
end
```

Pour vérifier qu'il n'y a pas d'erreur dans la fonction, on tape par exemple :
>> [A, b, x, msg]= GenereSysteme(10,14,5)

Exercice 3.

Soit A ϵ $R^{n\times n}$. On introduit le vecteur p_L^A (resp. p_C^A), appartenant à R^n, des indices des colonnes (resp. des lignes) du premier coefficient non nul de chaque ligne (resp. de chaque colonne). Par convention, si tous les coefficients d'une ligne (resp. d'une colonne) sont nuls, le nombre reporté est n + 1. Par exemple, pour la matrice :

$$A = \begin{pmatrix} 5 & 6 & 3 & 0 \\ 0 & 5 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 4 \end{pmatrix} \in \mathbb{R}^{4\times 4}, \text{on a } p_L^A = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \end{pmatrix} \text{ et } p_C^A = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 4 \end{pmatrix}$$

1. Écrire une fonction qui pour toute matrice calcule renvoie les vecteurs p_L^A (resp. p_C^A)
2. Vérifier votre fonction avec la matrice A.

Réponses :

1.- Créer un fichier profil.m
   - Y mettre le code suivant :

```
function [pL,pC] = profil(A)
%PROFIL crée le profil de la matrice A
N = size(A,1);
pL = zeros(N,1);
pC = zeros(N,1);
for i = 1:N
fL = min(find(A(i,:)));
fC = min(find(A(:,i)));
if isempty(fL) % ce test est nécessaire car quand tous les termes de la ligne sont nuls, find renvoie un résultat vide et non pas la valeur 0
    pL(i) = N+1;
else
    pL(i) = fL;
end

if isempty(fC)
    pC(i) = N+1;
else
    pC(i) = fC;
end
end
```

2. Pour vérifier que notre fonction renvoie bien le bon résultat, on teste avec la matrice A :

```
>> A = [ 5 6 3 0; 0 5 -1 0; 0 0 0 0; 0 0 -1 4 ] ;
>> [pL, pC] = profil(A)

pL =

     1
     2
     5
     3


pC =

     1
     1
     1
     4
```

On retrouve bien le bon résultat.